

Chapter 1: Introduction

Contents

- What Operating Systems Do
 - Computer-System Organization
 - Computer-System Architecture
 - Operating-System Structure
 - Operating-System Operations
 - Process Management
 - Memory Management
 - Storage Management
 - Protection and Security
 - Computing Environments
 - Open-Source Operating Systems
- 운영체제가 하는 일.



1.1 What Operating Systems Do

- Depends on the point of view ⇒ 사용자의 관점과 시스템의 관점으로 본다.
- **Users** want convenience, **ease of use**; don't care about **resource utilization** ⇒ 사용자가 자원을 걱정 (PC)
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy ⇒ 자원의 이용 효율을 극대화
- Users of dedicated systems such as **workstations** have dedicated resources but frequently use shared resources from **servers** ⇒ 자원의 사용과 효율성 증가가 필요.
- Handheld computers are resource poor, optimized for usability and battery life ⇒ smart phone
- Some computers have little or no user interface, such as embedded computers in devices and automobiles

What Operating Systems Do (Cont'd)

=> System의 관점에서

- OS is a **resource allocator**
 - Manages all resources (CPU, 메모리, ...)
 - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
 - Controls execution of programs to prevent errors and improper use of the computer

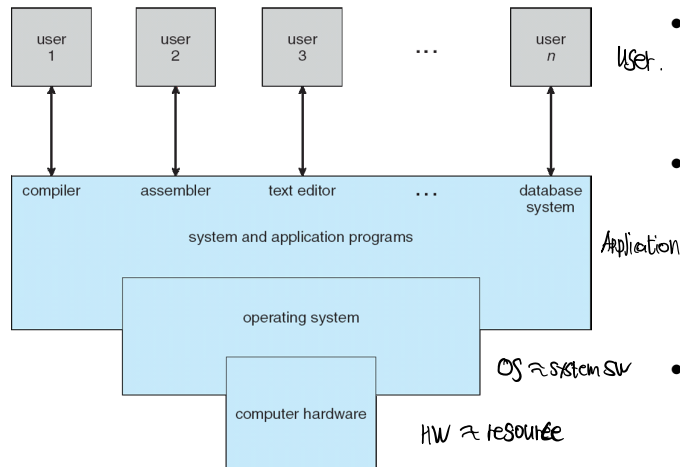
Operating System Definition

- No universally accepted definition
- “Everything a vendor ships when you order an operating system” is good approximation ⇒ 그렇다면 메카닉, 익스플로러 등?
- “The one program running at all times on the computer” is the **kernel**.
Everything else is either a system program (ships with the operating system) or an **application program**.
- A program that acts as an intermediary between a user of a computer and the computer hardware 중간자의 역할
- Operating system goals: ⇒ 운영체제의 목표.
 - Execute user programs and make solving user problems easier
 - Make the computer system convenient to use
 - Use the computer hardware in an efficient manner

1.2 Computer System Structure

구성요소들

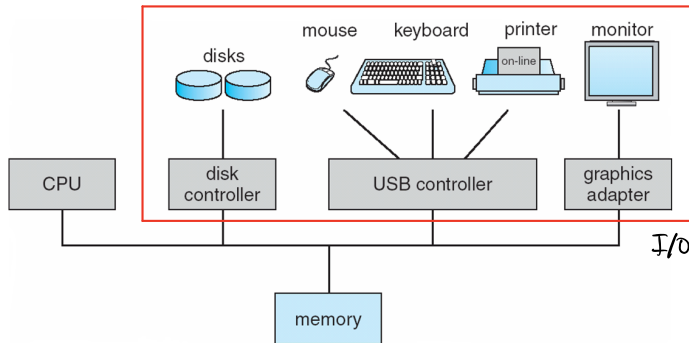
- Computer system can be divided into four components:



- Hardware** – provides basic computing resources
 - CPU, memory, I/O devices
- Operating system**
 - Controls and coordinates use of hardware among various applications and users
- Application programs** – define the ways in which the system resources are used to solve the computing problems of the users
 - Word processors, compilers, web browsers, database systems, video games
- Users** - People, machines, other computers

Computer System Organization

⇒ Computer hardware.



- Computer-system operation
 - One or more CPUs, device controllers connect through common bus providing access to shared memory
 - Concurrent execution of CPUs and devices competing for memory cycles 한 번에 1개 밖에 실행X.

Computer-System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type;
Each device controller has a local buffer
= CPU의 register
- CPU moves data from/to main memory to/from registers
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an **interrupt** ⇒ I/O가 CPU에게 보내는 중단 요청 신호.
⇒ 동작과정을 위한 방법.

Common Functions of Interrupts

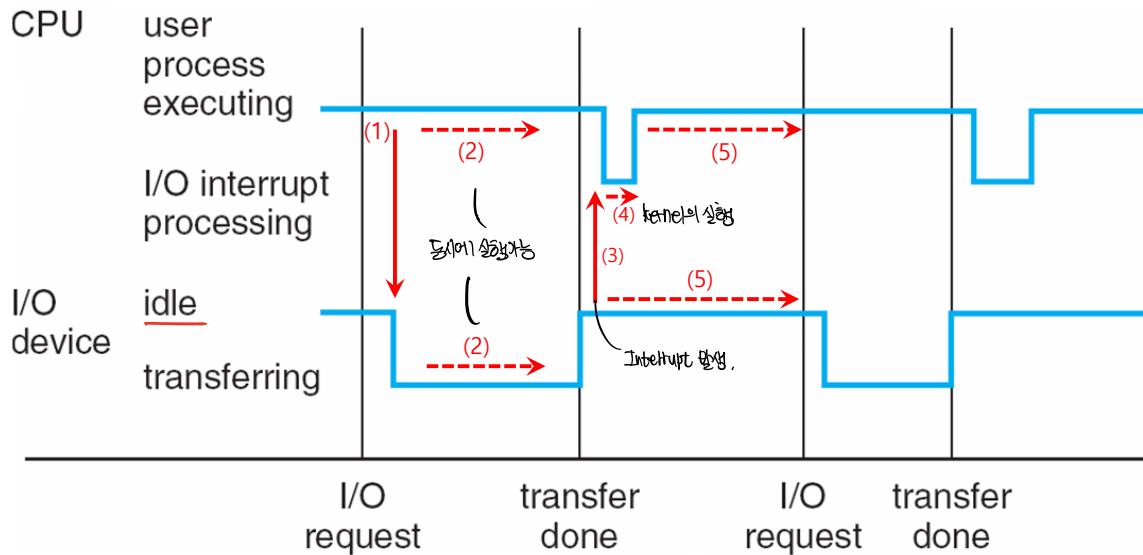
CPU는 라틴어를 참조하고 이동해서 처리한다.

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines ↳ interrupt을 나타내는 주소로 연결함 (주소 기록부)
- Interrupt architecture must save the address of the interrupted instruction HW로 보충함 interrupt
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request ⇒ 내부에서 발생함 (내부 CPU)
- An operating system is **interrupt driven** 외부 등장한다 받는다.

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter = PC (프로그램의 위치)
- Determines which type of interrupt has occurred:
 - **polling** → 모든 장치에게 interrupt가 발생했는지 물어보기.
 - **vectored** interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt

Interrupt Timeline

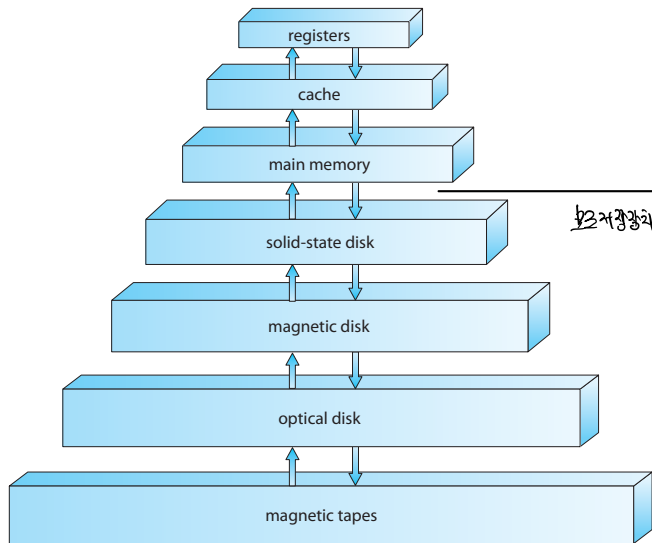


Storage Structure

- Main memory – only large storage media that the CPU can access directly
 - **Random access**
 - Typically **volatile** (휘발성) ⇒ 전원이 끊기면 데이터가 사라진다.
- Secondary storage – extension of main memory that provides large **non volatile** storage capacity
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material =(HDD)
 - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
 - The **disk controller** determines the logical interaction between the device and the computer
- **Solid-state disks** – faster than magnetic disks, nonvolatile =(SSD)
 - Various technologies
 - Becoming more popular

Storage Hierarchy

⇒ 저장장치의 계층구조



- Storage systems organized in hierarchy

- Speed 위조고속으로 읽음
- Cost "
- Volatility 보조저장장치에서 비휘발성

- Caching** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage 저속의 일차저장장치에 고속의 장치에 저장 ⇒ 성능향상

- Device Driver** for each device controller to manage I/O

- Provides uniform interface between controller and kernel

Caching

성능향상 기법.

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
⇒ 주어진 상황에서 최대한 빠르게 data에 접근하는 것.
- Faster storage (cache) checked first to determine if information is there
 - If it is, information used directly from the cache (fast) cache에 있다면 사용
 - If not, data copied to cache and used there → 없다면 복제하여 사용.
- Cache smaller than storage being cached
 - Cache management important design problem
 - Cache size and replacement policy
어떤 조건을 택지 교체하는지에 대한 방법.

⇒ 일특 data 빈 접근이 발생할 수 있다.

I/O Structure 2가지의 방식.

1. 첫번째

- After I/O starts, control returns to user program only upon I/O completion
 - Wait instruction idles the CPU until the next interrupt
 - Wait loop (contention for memory access) idle 명령.
 - At most one I/O request is outstanding at a time, no simultaneous I/O processing

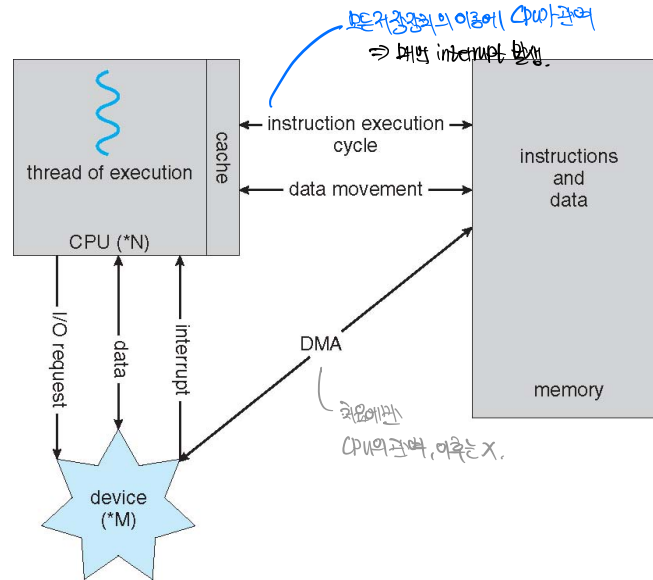
2. 두번째

- After I/O starts, control returns to user program without waiting for I/O completion ⇒ 동시에 여러일을 처리가능
 - **Device-status table** contains entry for each I/O device indicating its type, address, and state 를 유지한다.
 - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt

Direct Memory Access Structure (고속으로 동작하는 장치에서 사용)

- Used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block, rather than the one interrupt per byte

How a Modern Computer Works

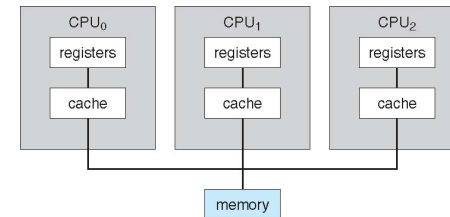


A von Neumann architecture
= Stored program

1.3 Computer-System Architecture

75% 이하

- Most systems use a single general-purpose processor (PDAs through mainframes) ⇒ single processor system
- **Multiprocessors** systems growing in use and importance
 - Also known as parallel systems, multi core system
 - Advantages include:
 1. **Increased throughput** 처리량의 증가 (processor가 많아지면)
 2. **Economy of scale** 경제성
 3. **Increased reliability – graceful degradation or fault tolerance** 신뢰성의 증가.
 - Two types:
 1. **Asymmetric Multiprocessing** / 매가 master
 2. **Symmetric Multiprocessing** 모든 node가 동등. (SMP)
 - **UMA** and **NUMA** variations
 - ↓
 - 모든 프로세서에 메모리 접근이 동일 but 동일하지 않은 메모리 접근 시간.



Symmetric Multiprocessing Architecture

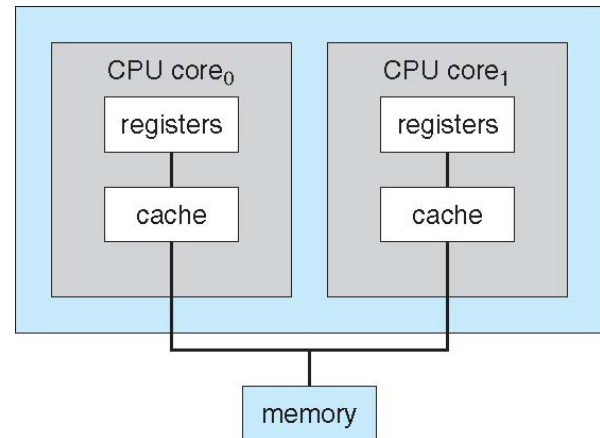
A Dual-Core Design

⇒ 하나의 chip에 여러 core를 넣는 것.

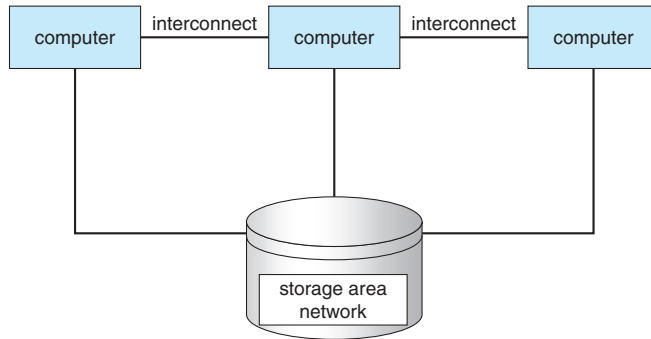
- Multi-chip and **multicore**

chip간 통신과 core간 통신이 복잡하다.

- Systems containing all chips vs. blade servers
 - Chassis containing multiple separate systems



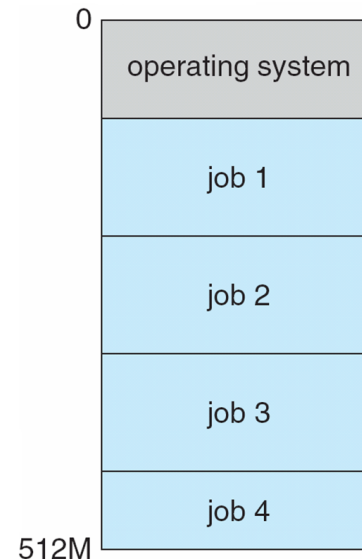
Clustered Systems



- Like multiprocessor systems, but multiple systems working together
 - Usually sharing storage via a **storage-area network (SAN)**을 통해 공유
 - Provides a **high-availability** service which survives failures
 - **Asymmetric clustering** vs. 한 node가 master.
 - **Symmetric clustering** 모든 node가 동등.
 - Some clusters are for **high-performance computing (HPC)**
 - Applications must be written to use **parallelization**

1.4 Operating System Structure

- **Multiprogramming** needed for efficiency
 - A subset of total jobs in system is kept in memory
 - Multiprogramming organizes jobs (code and data) so CPU always has one to execute. When it has to wait to another job
 - (for I/O for example), OS switches
 - job selected and run via **job scheduling**



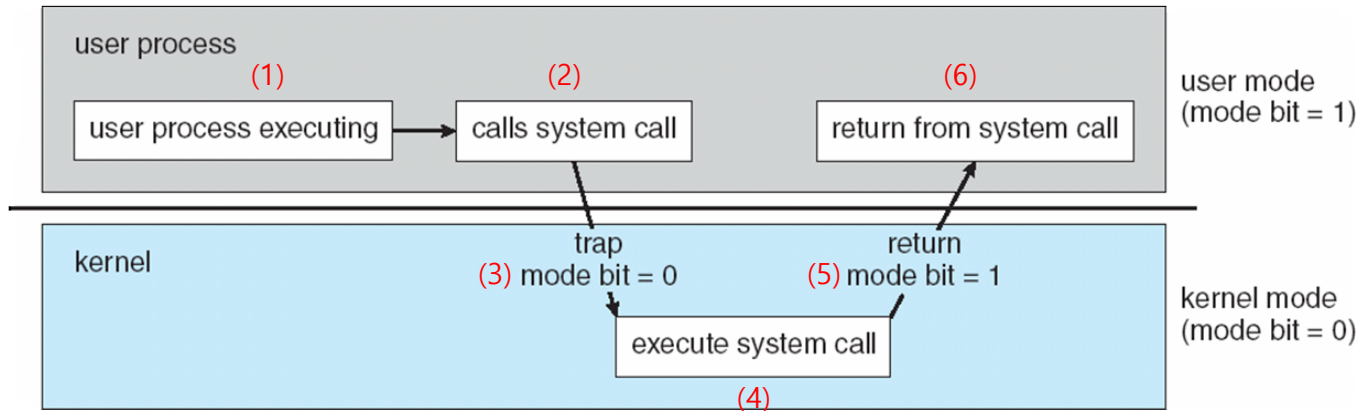
Timesharing

- **Timesharing** (**multitasking**) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
 - **Response time** should be < 1 second
 - Each user has at least one program executing in memory \Rightarrow **process**
 - If several jobs ready to run at the same time \Rightarrow **CPU scheduling**
 - If processes don't fit in memory, **swapping** moves them in and out to run
 - **Virtual memory** allows execution of processes not completely in memory

1.5 Operating-System Operations

- **Interrupt driven** by hardware
 - Software error or request creates **exception** or **trap** - Division by zero, request for operating system service
- **Dual-mode** operation allows OS to protect itself and other system components
 - **User mode** and **kernel mode**
 - **Mode bit** provided by hardware
 - Provides ability to distinguish when system is running user code or kernel code
 - Some instructions designated as **privileged**, only executable in kernel mode
 - System call changes mode to kernel, return from call resets it to user

Transition from User to Kernel Mode



1.6 Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a ***passive entity***, process is an ***active entity***.
- Process needs resources to accomplish its task
 - CPU, memory, I/O, files
 - Initialization data
- Process termination requires reclaim of any reusable resources
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
 - Concurrency by multiplexing the CPUs among the processes

Process Management Activities

- The operating system is responsible for the following activities in connection with process management:
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

1.7 Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
 - Optimizing CPU utilization and computer response to users
- Memory management activities
 - Keeping track of which parts of memory are currently being used and by whom
 - Deciding which processes (or parts thereof) and data to move into and out of memory
 - Allocating and deallocating memory space as needed

1.8 Storage Management

- OS provides uniform, logical view of information storage
 - Abstracts physical properties to logical storage unit - **file**
 - Each medium is controlled by device (i.e., disk drive, tape drive)
 - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
 - Files usually organized into directories
 - Access control on most systems to determine who can access what
 - OS activities include
 - Creating and deleting files and directories
 - Primitives to manipulate files and dirs
 - Mapping files onto secondary storage
 - Backup files onto stable (non-volatile) storage media

Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time. Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
 - Free-space management
 - Storage allocation
 - Disk scheduling
- Some storage need not be fast
 - Tertiary storage includes optical storage, magnetic tape
 - Still must be managed – by OS or applications

Performance of Various Levels of Storage

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

- Movement between levels of storage hierarchy can be explicit or implicit

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
 - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
 - Drivers for specific hardware devices
 - General device-driver interface

1.9 Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
 - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
 - User identities (**user IDs**, security IDs) include name and associated number, one per user
 - User ID then associated with all files, processes of that user to determine access control
 - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
 - **Privilege escalation** allows user to change to effective ID with more rights

1.11 Computing Environments - Traditional

- Stand-alone general purpose machines
- But blurred as most systems interconnect with others (i.e. the Internet)
- **Portals** provide web access to internal systems
- **Network computers** (**thin clients**) are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming ubiquitous – even home systems use **firewalls** to protect home computers from Internet attacks

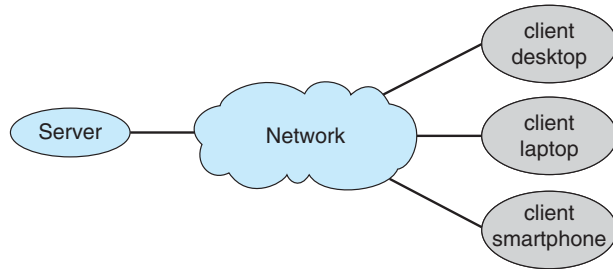
Computing Environments - Mobile

- Handheld smartphones, tablets, etc
- What is the functional difference between them and a “traditional” laptop?
- Extra feature – more OS features (GPS, gyroscope); Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**

Computing Environments – Distributed

- Collection of separate, possibly heterogeneous, systems networked together
- **Network** is a communications path, **TCP/IP** most common
 - **Local Area Network (LAN)**
 - **Wide Area Network (WAN)**
 - **Metropolitan Area Network (MAN)**
 - **Personal Area Network (PAN)**
- **Network Operating System** provides features between systems across network
 - Communication scheme allows systems to exchange messages
 - Illusion of a single system

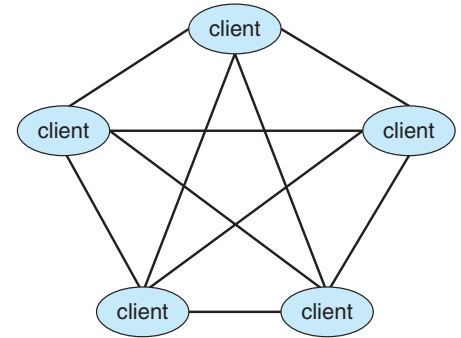
Computing Environments – Client-Server



- Dumb terminals supplanted by smart PCs
- Many systems now servers, responding to requests generated by clients
 - Compute-server system provides an interface to client to request services (i.e., database)
 - File-server system provides interface for clients to store and retrieve files

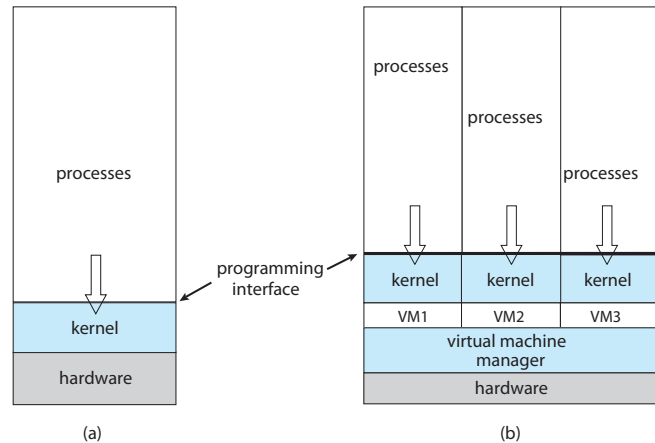
Computing Environments - Peer-to-Peer

- Another model of distributed system
- P2P does not distinguish clients and servers
 - Instead all nodes are considered peers
 - May each act as client, server or both
 - Node must join P2P network
 - Registers its service with central lookup service on network, or
 - Broadcast request for service and respond to requests for service via **discovery protocol**
 - Examples include Napster and Gnutella, **Voice over IP (VoIP)** such as Skype



Computing Environments - Virtualization

- Allows operating systems to run applications within other OSes
- **Emulation** used when source CPU type different from target type (i.e. PowerPC to Intel x86)
 - Generally slowest method
 - When computer language not compiled to native code – **Interpretation**
- **Virtualization** – OS natively compiled for CPU, running **guest** OSes also natively compiled
 - Consider VMware running WinXP guests, each running applications, all on native WinXP **host** OS
 - **VMM** provides virtualization services

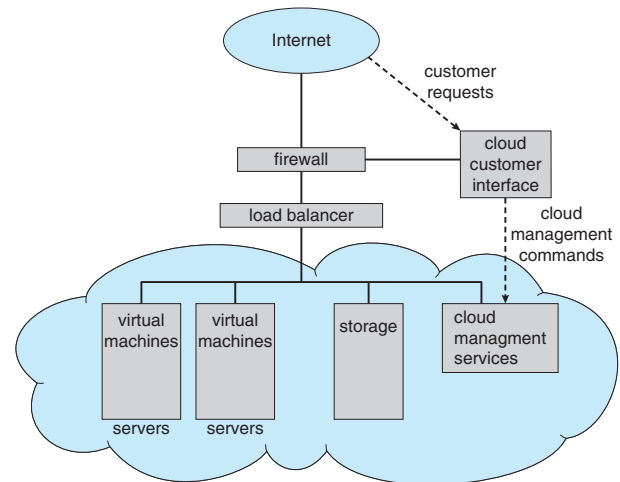


Computing Environments – Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization as based on virtualization
 - Amazon **EC2** has thousands of servers, millions of VMs, PBs of storage available across the Internet, pay based on usage
- Many types
 - **Public cloud** – available via Internet to anyone willing to pay
 - **Private cloud** – run by a company for the company's own use
 - **Hybrid cloud** – includes both public and private cloud components
 - Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e. word processor)
 - Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e. a database server)
 - Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e. storage available for backup use)

Computing Environments – Cloud Computing

- Cloud compute environments composed of traditional OSes, plus VMMs, plus cloud management tools
 - Internet connectivity requires security like firewalls
 - Load balancers spread traffic across multiple applications



Computing Environments – Real-Time Embedded Systems

- Real-time embedded systems most prevalent form of computers
 - Vary considerable, special purpose, limited purpose OS, **real-time OS**
 - Use expanding
- Many other special computing environments as well
 - Some have OSES, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
 - Processing ***must*** be done within constraint
 - Correct operation only if constraints met